**Polimorfizm dynamiczny, funkcje wirtualne – sprawdzian – grupa B**

Określ kilka różnych figur geometrycznych, tj. Trójkąt równoboczny, Pięciokąt foremny, Sześciokąt foremny dziedziczących po klasie bazowej Bryła. Każda bryła musi udostępniać metodę double obwod() (virtual w klasie Bryła), zwracającą **obwod bryły**. Ma to umożliwić np. iterowanie po tablicy różnych brył (czyli tablicy zmiennych typu Bryła \*) celem obliczenia sumy obwodów wszystkich brył w tej tablicy. Dodaj klasę Wielokąt (oczywiście dziedziczącą po class Bryla), pamiętającą listę punktów na płaszczyźnie, które połączone kolejno odcinkami tworzą wielokąt. Należy zadbać o poprawne zwalnianie pamięci po obiektach tej klasy.

Utwórz obiekty w/w klas, wyświetl na ekranie obwody tych brył oraz sumę o obwodów wszystkich brył.

Oto implementacja zgodna z wymaganiami:

#include <iostream>

#include <cmath>

#include <vector>

class Bryla {

public:

virtual double obwod() const = 0;

virtual ~Bryla() {}

};

class FiguraGeometryczna : public Bryla {

public:

virtual ~FiguraGeometryczna() {}

};

class Wielokat : public Bryla {

private:

std::vector<std::pair<double, double>> punkty;

public:

Wielokat(const std::vector<std::pair<double, double>>& p) : punkty(p) {}

double obwod() const {

double obwod = 0;

int n = punkty.size();

for (int i = 0; i < n; i++) {

double x1 = punkty[i].first;

double y1 = punkty[i].second;

double x2 = punkty[(i + 1) % n].first;

double y2 = punkty[(i + 1) % n].second;

obwod += std::sqrt(std::pow(x2 - x1, 2) + std::pow(y2 - y1, 2));

}

return obwod;

}

};

class TrojkatRownoboczny : public FiguraGeometryczna {

private:

double dlugoscBoku;

public:

TrojkatRownoboczny(double dlugosc) : dlugoscBoku(dlugosc) {}

double obwod() const {

return 3 \* dlugoscBoku;

}

};

class PiekatokatForemny : public FiguraGeometryczna {

private:

double dlugoscBoku;

public:

PiekatokatForemny(double dlugosc) : dlugoscBoku(dlugosc) {}

double obwod() const {

return 5 \* dlugoscBoku;

}

};

class SzesciokatForemny : public FiguraGeometryczna {

private:

double dlugoscBoku;

public:

SzesciokatForemny(double dlugosc) : dlugoscBoku(dlugosc) {}

double obwod() const {

return 6 \* dlugoscBoku;

}

};

int main() {

TrojkatRownoboczny trojkat(5);

PiekatokatForemny pieciokat(4);

SzesciokatForemny szesciokat(6);

std::vector<Bryla\*> bryly;

bryly.push\_back(&trojkat);

bryly.push\_back(&pieciokat);

bryly.push\_back(&szesciokat);

double sumaObwodow = 0;

for (const auto& bryla : bryly) {

sumaObwodow += bryla->obwod();

}

std::cout << "Obwód trójkąta równobocznego: " << trojkat.obwod() << std::endl;

std::cout << "Obwód pięciokąta foremnego: " << pieciokat.obwod() << std::endl;

std::cout << "Obwód sześciokąta foremnego: " << szesciokat.obwod() << std::endl;

std::cout << "Suma obwodów wszystkich brył: " << sumaObwodow << std::endl;

// Zwolnienie pamięci

for (const auto& bryla : bryly) {

delete bryla;

}

return 0;

}